(2)

physical dynamics, inc.

# RES OPERATIONS

RES-FR-009-90



FINAL REPORT

SSBN TACTICAL SECURITY EXERCISE SIMULATOR

AND

TACTICAL DEVELOPMENT COMPUTER PROGRAM

CONTRACT #N00014-87-C-0063

8 MAY 1990

SUBMITTED BY:

PHYSICAL DYNAMICS, INC.
RES OPERATIONS
P. O. BOX 9505
ARLINGTON, VA 22209

RES-FR-009-90

# FINAL REPORT

## SSBN TACTICAL SECURITY EXERCISE SIMULATOR

## AND

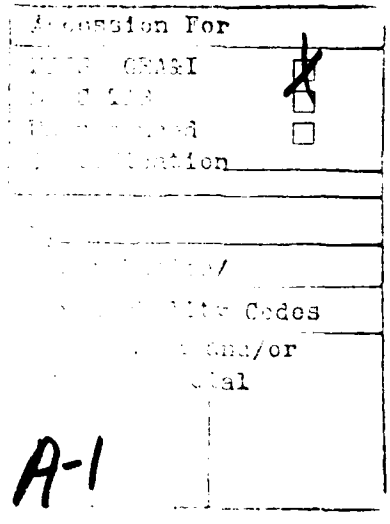
## TACTICAL DEVELOPMENT COMPUTER PROGRAM

NTRACT #N00014-87-C-0063

8 MAY 1990

SUBMITTED BY:

PHYSICAL DYNAMICS, INC.
RES OPERATIONS
P. O. BOX 9505
ARLINGTON, VA 22209




UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS<br>NONE |
| 2a. SECURITY CLASSIFICATION AUTHORITY<br>NOT APPLICABLE | 3 DISTRIBUTION / AVAILABILITY OF REPORT<br>UNLIMITED |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE<br>NOT APPLICABLE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S)<br>RES-FR-009-90 | 5 MONITORING ORGANIZATION REPORT NUMBER(S)<br>NONE |
| 6a. NAME OF PERFORMING ORGANIZATION<br>PHYSICAL DYNAMICS, INC.<br>RES OPERATIONS<br>6b OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>OFFICE OF NAVAL RESEARCH |
| 6c. ADDRESS (City, State, and ZIP Code)<br>1601 N. KENT STREET, #1100<br>ARLINGTON, VA 22209 | 7b. ADDRESS (City, State, and ZIP Code)<br>800 N. QUINCY STREET<br>ARLINGTON, VA 22217-5000 |
| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION<br>OFFICE OF NAVAL RESEARCH<br>8b. OFFICE SYMBOL (If applicable)<br>1511A:RMD | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>R&T # 220h048---01/8-08-86 (122) |
| 8c. ADDRESS (City, State, and ZIP Code)<br>Same as Item 7b. | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. Same as 9. / TASK NO. / WORK UNIT ACCESSION NO. |

11 TITLE (Include Security Classification)
SSBN TACTICAL SECURITY EXERCISE SIMULATOR AND TACTICAL DEVELOPMENT COMPUTER PROGRAM

12 PERSONAL AUTHOR(S)
CAPT Gerald E. Green, USN(Ret.) and Dr. Gregory Korzeniewski

| 13a. TYPE OF REPORT | 13b TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| FINAL TECHNICAL | FROM NOV 86 TO APR 90 | 1990 MAY 8 | 18 |

16 SUPPLEMENTARY NOTATION
NONE

| 17 COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | TACTICAL SECURITY, EXERCISE SIMULATOR, TACTICAL DEVELOPMENT, COMPUTER PROGRAM |
| | | | |
| | | | |

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The essence of this work was the creation and refinement of a user-friendly computer simulation for submarine-on-submarine engagements.

| 20 DISTRIBUTION / AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | UNCLASSIFIED |

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| CAPT Gerald E. Green, USN(Ret.) | (703) 276-1300 | RES Operations |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

Synopsis

This report describes work completed under Contract #N00014-87-C-0063. The essence of this work was the creation and refinement of a user-friendly computer simulation for submarine-on-submarine engagements.

I. Introduction and Discussion.

Simulations are assuming an ever more important role in the life of weapon systems, from defining and evaluating early development options to achieving the most effective employment methods/tactics. Their use goes far beyond even these applications, as realistic simulations are utilized to develop operationally oriented data to support high level decision making, e.g., employing simulations to evaluate plans from battle group tactics to general war plans. As the operating costs for various tactical exercises soar, the relatively inexpensive application of simulations has gained in popularity. While simulations cannot replace all at-sea evolutions, there is a clear area where their use is efficient and effective.

Closer to home, RES Operations was involved in a number of projects in support of a number of government agencies which required the evaluation of various factors, e.g., effectiveness and value-added; tasks well suited to the application of simulations. These evaluations could best be accomplished by utilizing periodic operator intervention during a series of excursions which portrayed various platforms and sensors operating in accordance with approved and accepted tactical doctrine.

An examination of existing Navy models resulted in the conclusion that each had shortcomings when applied to the specific tasks undertaken by RES Operations. These shortcomings ranged from being very programmer intensive when changing key factors during an excursion to an inability to observe a tactical engagement unfolding. This last feature proved to be much more important than originally anticipated and only now are we coming to fully appreciate the magnitude of the advantage of being able to see the interaction occur instead of being limited to examining a pile of relatively sterile statistics.

With these requirements and realizations as a basis, RES Operations set out to develop its own simulation incorporating many of the desired features found wanting in the available models. This was to be an object (platform, sensor, environment, etc.) oriented, rule (doctrine) based, operator interactive system programmed in LISP and running on a Symbolics 3600 series computer. (The program was subsequently configured to run on a MacIntosh containing a Symbolics 'MacIvory' board)

The basic makeup of the simulation has been described in a number of earlier publications, including the previous Progress Report. A more detailed description is included as Appendix A followed by a block diagram.

This paper will concentrate on addressing the work accomplished in this contract effort, subsequent improvements, and future implications.

II. Work Accomplished.

The contract effort, initiated in 1986 and subsequently modified in 1988, was directed at developing and demonstrating a versatile, operator-interactive submarine-on-submarine computer simulation. The 1988 modification focused the ORBIS development effort more sharply on the operational and tactical problems associated with potential hostile encounters between SSBNs and adversary SSNs.

Accomplishments in developing a computer simulation during this contract will be addressed by general area:

Rule Editor

A package of rules (called "rule base systems") that guide the operation of objects in the ORBIS model is in place and completely accessible to the non-programming operator through an interactive menu-driven rule editor. Rules are presented in editor windows in an English-like syntax, along with a dictionary of terms and a menu choice of operations. Rules can be added, edited and/or removed with a few menu selections made with a mouse.

The rule editor allows changes of an object's behavior to be made at any point in the simulation: at the start of the simulation run, or at any point during the course of

the simulation. The effect of the rule change is immediate and dynamic; the object's behavior is guided by the new rule from the point of change with no adverse effect on the other parts of the program. The rule editor permits changes to be temporary or permanent, and warns the user before any changes are made permanent in order to prevent inadvertent changes to the program.

Object Oriented Program

Objects (such as platforms, sensors, weapons, mines, etc.) are represented in ORBIS as independent data structures that embody all the information required to model their interaction in a tactical engagement. The object-oriented programming design of ORBIS means that objects have associated with them not only data, but any and all functions and algorithms specific to their operation in the model, as well.

Object-oriented design enables the user to combine objects, for example, a submarine, spherical array, a towed array and an active sensor into a accurate representation of a submarine with a complete sensor suite. ORBIS objects are interchangeable modules that

can be attached, switched and removed without concomitant changes in source code programming. An AKULA-class submarine can be given a towed array, a 688-class can operate employing Soviet tactics and fire ET-80a torpedoes, or a mine-hunting sub can be given a tethered UUV--all by attaching or removing modular objects through a simple series of menu choices.

## Architecture Design

The ORBIS simulator is a powerful tactical development tool due to its fundamental design as an object-oriented, rule-based interactive simulator. The time-stepped simulation is controlled by an operator using interactive editing tools and a graphic display of the engagement. Data parameters and rules have been obtained through accepted Navy sources and reviewed for accuracy by experienced submarine commanders. Environmental data, sound propagation, fluctuation models and sonar detection models have been obtained from accepted Navy sources and models (e.g., SFMPL, RAYMODE, SIM II, etc.).

The operator has complete access to all the rules involved in a simulation, the selection of the

environmental conditions in the engagement and the participants along with their complement of sensors, weapons and countermeasures. The operator can select a variety of predefined MOEs, set conditions for the dynamic removal of an object from the simulation (such as lifetime of a CM or torpedo), control under what conditions the simulation completes a run.

As an alternative to interactive runs, ORBIS can be run in batch (or "background") mode in which multiple runs of the simulation are used to gather statistical data. Each background run is stored in such a manner that if its results merit a closer look, the run can be reproduced in interactive mode.

Inclusion of Accepted Data/Algorithms and Run Time

Most ORBIS simulation runs that examine SSBN tactical security issues such as hostile encounter are completed in 3-5 minutes.

Realistic Rule Bases for Hostile Encounter

ORBIS has a complete set of rule base systems for the SSBN hostile encounter scenario, including U.S. SSBN

Hostile Encounter, Soviet SSN Hostile Encounter, U.S. and Soviet Countermeasure Deployment, U.S. and Soviet Torpedo Fire, MK-48 Search, Attack and Reattack as well as a postulated rule base for Soviet ET-80a torpedoes.

Enhancements include upgrade from the initially-requested geometric approximation torpedo detection method to a rule base system utilizing an active sensor on the torpedo object.

Upgraded Object Characteristics Modules

Upgrades include the incorporation of the RAYMODE model to produce propagation loss data (replacing the SFMPL prop loss data), acquisition of aspect-dependent noise signature data and incorporating it into the submarine objects, creation of platform-specific submarine objects, enhancement of the submarine kinematic model (using advance, transfer, acceleration and dive-rate data from David Taylor).

Simulation Validation

Reproduced the initial conditions of the security exercise PAC 2-87 Phase III which examined the

survivability of a TRIDENT in a hostile encounter engagement. The four at-sea runs were reproduced with the ORBIS simulator with equivalent initial detection ranges. Additionally, numerous excursions were run designed to enhance the realism that is lost in at-sea exercises (depth separation, use of UWT, no countermeasures).

## III. FUTURE/IMPLICATIONS

The work accomplished under this contract demonstrated the versatility and flexibility of this approach to simulation architecture. The result is an application which is useful in accurately depicting the sub-on-sub hostile encounter. This effort in effect provided an operating model which serves as the basis for the subsequent (ongoing) improvements sponsored by OP-213. The end result will be an analytical tool ideally suited to the upcoming examination for APL of SSBN tactical considerations and security exercises.

# APPENDIX A

# ORBIS SIMULATION TECHNIQUE

## 1.1 INTRODUCTION

ORBIS (Object-oriented Rule-Based Interactive System) is a general purpose software based development and analysis tool. It was designed to provide a capability for simulation, analysis, or control in problems in which real time data flow must be analyzed and in which actions must be taken which depend on that analysis. Its internal architecture allows it to be used to consider a wide variety of applications ranging from warfare simulation to autonomous vehicle control to power plant fault analysis. For example, ORBIS has been used in development of tactics for both Submarine and Anti-submarine Warfare, technical evaluations of potential system enhancements in these warfare areas, and in verification of critical control logic for unmanned underwater vehicles.

Interfaces consisting of special purpose editors, menus, graphics, and a time-stepped control structure allow the user to interactively manipulate all relevant aspects of the problem under consideration including the basic logic which is used in a particular application. The scope, flexibility, and interactive

features of ORBIS make it possible to reduce development time and to involve domain experts, decision makers, and others without extensive programming experience in the development and analysis process.

ORBIS is currently implemented on a SYMBOLICS computer and is written in Zeta-lisp. Certain applications have taken advantage of the capability to employ multiple language source mode modules (FORTRAN 77 - LISP).

## 1.2 SYSTEM OVERVIEW

This section describes the overall ORBIS system. The basic ORBIS system software is comprised of four sophisticated, highly-integrated modules: The Main System, Rule-Editor, Dictionary-Editor, and Object-Tree-Editor. A complete ORBIS simulation requires these modules as well as input data files which define simulation rule bases, dictionary terms, objects, and setup parameters.

**MAIN SYSTEM**

The main system module controls the simulation and allows ORBIS to be run in either of two modes: interactive or background. In the interactive mode, the user has full control over the simulation as it progresses. An overhead graphics view and a status window allow the user to monitor, interrupt, modify,

restart, replay, and save changes to the simulation. This mode is especially useful in framing "what-if" questions and in answering "Why?". The background mode is designed to collect Monte-Carlo statistics by storing user-specified results in a file.

**RULE-EDITOR**

The rule editor is the means by which rule base systems are made and modified. Rules are made by splicing together the English representation of dictionary variables (discussed below). The rule editor displays current dictionary terms, logical constructs such as 'if', 'and', 'or' or '=', and algebraic functions. The rule editing windows are mouse-sensitive. The rules are constructed by simply positioning the mouse over the desired term and clicking. Once a rule is created it may be modified in any way. The rule editor may be accessed during a program run so that the effect of a rule change may be immediately assessed.

**DICTIONARY-EDITOR**

The dictionary editor provides the means by which needed terms and algorithms can be added to the system and by which all dependencies can easily be traced. The dictionary is a collection of terms, defined by the user, called dictionary variables which serve several important roles. First, they are the link between

rule bases and object attributes[1]. Second, they define the information which may be passed between rules. Third, they serve as variables in rules which act on calculated value.

If a term is needed to make a rule which is not present in the rule editor windows, the dictionary editor may be called to add a new term. This may be done either before or during rule editing.

**OBJECT-THREE-EDITOR**

In the real world, an object is an entity that is visible or otherwise tangible. In ORBIS an object is any entity that acts or can be acted upon: in a sense, objects in ORBIS parallel objects from the real world. Furthermore, as in the real world, objects exist in time and hence can be created, destroyed, copied, shared, and updated.

**RULE BASES**

Closely associated with the objects are rule bases which determine how, when, and under what circumstances the objects change their attributes. In essence, a rule base is a collection of statements which take the form IF {condition} THEN {action}. These condition and action statements may be quite complex,

---

[1]Each object has one or more attributes. Examples of attributes for an object of type submarine might include course and speed. Rules do not directly read or set the values of attributes. Rather, rules access the value of dictionary variables such as ordered-course, current-course, ordered-speed, and current-speed.

involving operators such as AND, OR, and UNTIL. Rules are aggregated, in hierarchical fashion, to form a rule base system.

Objects can be associated with other objects and with rule bases; for example, an object of submarine type can be associated with various sensor objects (active sonar, passive towed array), weapon objects (torpedo), and rule bases (search, torpedo evasion). Objects and rule base systems are grouped together to form an object tree.

A collection of object-trees, together with initial values, time increment conditions, statistics, and choices for other parameters and variables is called a setup. With ORBIS, the user can start with what is essentially a 'blank sheet' and make a setup from existing objects and rule base systems.

When a setup is made, it is also possible to choose specific geographic locations from a data base of contour maps. Since the treatment of this data base is also modular, new maps can be easily entered.

## 1.3 OVERVIEW OF ORBIS INTERACTIVE FEATURES

One of the unique features of ORBIS is its on-line application development and analysis aids. These allow a user to interactively modify all of the program components both in the development phase and at run time.

**RULE BASE EXAMINER**

The rule base examiner displays rule base systems to users. All rules appear in an English-like language. This, together with a cursory knowledge of the way in which the rules are invoked, allows experts or users without programming experience to examine, and using the rule editor, to modify the logic present in the program. ORBIS also includes a 'hardcopy' option which will print the English representation of rule base systems to hardcopy.

**OBJECT CREATION AND DESTRUCTION**

ORBIS also contains features so that objects can be interactively created or destroyed. That is, objects not present at the beginning of a run can be added, or conversely, objects present can be removed. For example, torpedoes can be launched to destroy ships, helicopters can be launched from surface ships, sonobuoys can be dropped from the helicopters, and large distributed sensor fields with many sensors can be modeled.

**VARIABLE TIME STEP**

The ORBIS control structure is time-stepped, with an option to control the time increment with event dependency. All aspects of this structure can be interactively controlled by the user.

**INITIAL AND TERMINAL CONDITIONS**

When the program is run the user chooses whether to run the program in the interactive or the background mode. If the

background mode is chosen, the user selects from a list of functions which can randomize any initial object attribute or rule base variable, chooses the conditions that will end a run, chooses the statistics that will be kept, and chooses the numbers of runs. Results can be written to a file.

**FUNCTION REUSABILITY**

Time increment conditions, remove conditions, initialization functions, end conditions, and MOEs are all represented by LISP functions. These are created in a modular fashion so that they are all dependent, and once constructed, can easily be inserted and used with any setup.

**INTERACTIVE CONTROL**

When the program is run in the interactive mode, the program of the simulation is displayed for user "as it happens". At any point during the run the user may stop the simulation, invoke one or several editing and interactive control functions, and then continue the run. Some of these functions are: edit object attributes, move object, add new object tree, edit rules, view all active rules, restart, replay, and restore to an earlier state of the system. The ability to interrupt a run and to employ these features provides a powerful development and analysis environment. For example, a user can stop the simulation just past a critical point, restore the state at that point, invoke the rule editor to

change a rule, and then continue the simulation to immediately assess the affect of the changed rule. The ability to try any combinations of objects in object trees, and to change any of the objects' attribute values gives the user great flexibility in conducting technical assessments and value-added analyses.